John von Neumann Institute for Computing

Fault Tolerant Master-Worker over a Multi-Cluster Architecture

J. Rodriguez de Souza, E. Argollo, A. Duarte, D. Rexachs, E. Luque

published in

Parallel Computing:
Current & Future Issues of High-End Computing,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 465-472, 2006.

http://www.fz-juelich.de/nic-series/volume33

Fault-Tolerant Master-Worker over a Multi-cluster Architecture *

Josemar Souza[abc], Eduardo Argollo[a], Angelo Duarte[ab], Dolores Rexachs[a], Emilio Luque[a]

[a] CAOS - Computer Architecture and Operating Systems Department, Universitat Autònoma de Barcelona (UAB), 08193, Barcelona, Spain

[b] CEBACAD - High Performance Computation Center of Bahia, Universidade Catolica do Salvador (UCSAL), Av. Cardeal da Silva, 205, 40.220-140, Salvador, Bahia, Brazil

[c] Universidade do Estado da Bahia (UNEB), Rua Silveira Martins, 2555, 41.150-000l, Salvador, Bahia, Brazil

Abstract: The growth of clusters into cluster collections increases potential points of failures, requiring the implementation of a fault-tolerance scheme. The CoHNOW is organized as a hierarchical master-worker scheme and clusters may be geographically distributed and interconnected by Internet. This paper describes a system of Fault-Tolerant protection by Data Replication (FT-DR), based on preserving critical functions by on-line dynamic data replication. The system-model target is to detect failures in any of the system functional elements and to tolerate this failure by recovering system consistency, guaranteeing the completion of the work in progress (recovery procedure). The model is designed to tolerate more than one simultaneous failure. There are three distinct phases for model-fault tolerance activities: startup, normal execution including failure detection monitoring, and failure recovery. The system is oriented for general master-worker applications running on CoHNOW and is transparent both for user and application. The master-worker environment requirements to support all these capabilities and the runtime overhead are under evaluation.

1. Introduction

Joining geographically distributed dedicated heterogeneous networks of workstations (HNOW) through Internet can be an inexpensive approach to achieving data-intensive computation. A collection of HNOW (CoHNOW) can be efficiently organized for parallel-application execution as a hierarchical master/worker scheme where each cluster is a master/worker itself [5]. This kind of CoHNOW presents several potential points of failure, namely, each cluster computer, local area networks and Internet network interconnection. The lack of quality guarantees for any of these components and the possibility of wasting considerable computational effort requires the implementation of a fault-tolerance scheme.

The general approach to protecting distributed parallel applications is to implement some sort of checkpoint strategy [4]. Currently, certain research groups are investigating the Fault-Tolerance problem in parallel applications with distributed memory.

We have developed a model that provides functional fault tolerance for all clusters' hosts by means of data replication. The model's target is to detect failures in any of the system functional elements and to tolerate this failure by recovering system consistency, guaranteeing the completion of the work (recovery procedure). In our model, if more than one failure occurs during the system-recovery procedure, these failures are considered to be concurrent. The model can be configured to admit the possibility of tolerating a number "C" of concurrent failures.

*The first author is grateful to the Universitat Autònoma de Barcelona (UAB), the Universidade Catolica do Salvador (UCSAL) and Universidade do Estado da Bahia (UNEB), for academic and financial support given during this work.

The proposed system of fault-tolerant protection by data replication (FT-DR), based on preserving critical functions by on-line dynamic data replication, is oriented for general Master/Worker (MW) applications running on CoHNOW and is user and application transparent. The model corresponds to a middleware transparent to the user, implemented as a layer between the application and the library on the message passing MPI.

For our architecture, based on the hierarchical Master/Worker (MW) model (all workers run the same program and there is no need to checkpoint [3] workers), a data-replication approach should be less computing expensive (low overhead) because it avoids the checkpoint cost of replicating all program states (memory, opened files, and code itself) for all machines, as was proved by Weissman in [6].

Weissman does not carry out data replication; he undertakes replicas of complete function/task. Our approach is similar, because protective workers act as ”sleeping masters” that will be enabled to receive the copies of the data and ”then go back to sleep”. When a master fault is detected, a ”sleeping master” ”awakes” and is converted into an enabled master.

Weissman [6] explores two options: checkpoint-recovery, quantitatively, and wide-area replication, as the means of achieving fault tolerance. His experimental results suggest that checkpoint-recovery can be preferable for small problems, whilst replication is preferable for larger problems, i.e. precisely those that use WAN environments (as in our case). The results also show that it is possible to predict overheads for the two methods.

The master-worker environment requirements needed to support all the proposed model capabilities and the runtime overhead are under evaluation.

The following sections present the study in further detail. Section 2 describes the system architecture. Our Functional Model is explained in Section 3. In Section 4, the experiment results are shown, and finally, conclusions and further work are presented in Section 5.

2. System Architecture

To develop the Functional Model of Fault-Tolerance Cluster Computers based on wide-area data replication (FT-DR), this was conceived as a Master/Worker (MW) hierarchical architecture, as shown in Figure 1.

The hierarchical CoHNOW architecture presents four main logical functionalities: the main master (MMT) is the master (MT) in which the computation is started and where initially all data to be processed resides. The sub-cluster (SC) represents a remote cluster. The sub-master (SMT) computation is a subset of the main-master (MMT) problem and receives the data through the wide-area network link. To isolate the local network from the SC traffic and to provide a reliable transparent communication between MMT and SMT, the architecture presents the communication manager function (CM). Finally, the computational data is processed by the workers (WK) in all clusters. An SC is seen by MMT as a worker with great latency of communication and ”great” computation power.

The MW paradigm was chosen because it allows portability to a wide range of applications. In this architecture, any of the hosts of a CoHNOW can be a Main Master (MMT), Worker (WK), Communication Manager (CM), and Sub Master (SMT). In other words, any host of the CoHNOW can take over other functions and data of the CoHNOW in case of failure. However, cluster heterogeneity should be borne in mind when choosing the function of the hosts. In this architecture (hierarchical MW), for each Sub Cluster (SC, also called remote clusters), there will be a CM in the Main Cluster (MC, also called local HNOW).

The CM is responsible for maintaining a local MPI connection and a Socket connection with the

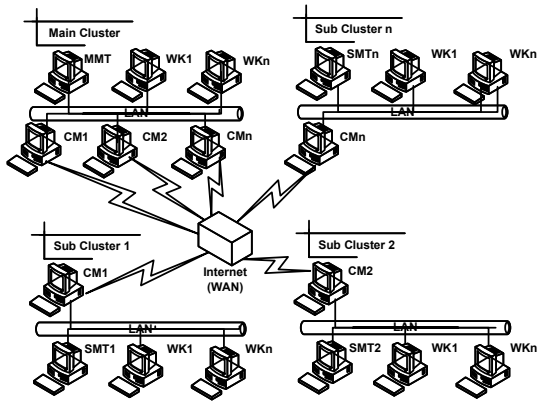

Figure 1. System Architecture

remote HNOW, therefore acting as a gateway for all incoming and outgoing data. The CM function can be a dedicated machine or a logical machine (a process running on a host of the cluster). All CoHNOW hosts have two network adapters. One of these network adapters is used to access the local network (NET) and the other is used to access the remote network (Internet). Access to the remote network (Internet) by one of the cluster hosts is given when that host takes over CM functions, in the case of failure in one CM [1].

This architecture is enhanced for fault-tolerant support with the usage of a developed fault-tolerant communication library by the CM functionalities. In order to support CM failures (transitory) and because the CM workstation needs two network cards (intra-cluster and inter-cluster), this two-network card facility is included in all workstations so that any computer can assume any functionality.

3. Functional model

In our model we use wide-area data replication, which is an important technique for increasing computer system availability of this. This occurs when a data set is copied and attributed to more than one unit of processing. This Fault-Tolerance technique (application) has an associated cost: wide-area data replication consumes computational and input/output resources, decreasing the performance of the system as a whole. In our model, Fault Tolerance does not imply physical redundancy; we are interested in ”functional redundancy”, in other words the functions and the data are taken over by other hosts of the cluster in case of failure in any component of the CoHNOW.

FT-DR provides Fault Functional Tolerance for all the elements of a cluster by means of Data Replication. The goal of the model is to detect faults in any of the functional elements within the system and to allow the fault to recover so as to ensure system consistency and to guarantee the conclusion of the work, despite performance loss. In our model, if more than one fault occurs during the system’s recovery procedure, these faults are considered as concurrent. The proposed model allows selecting the number of concurrent faults (”C”) - protection level - to be tolerated by the system. The model corresponds to a middleware transparent to the user. This is a layer between application and the library on the message passing MPI. (Figure 2).

In our model, the protected elements are: MMT, WK, CM, SMT and Network (NET). All failures in the CoHNOW hosts are considered permanent. Failure of the network (local or remote) is considered transitory, in other words, a failure can occur. However, we presuppose that the network will

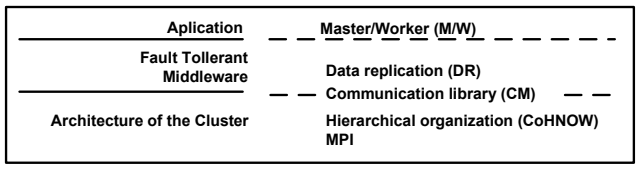

Figure 2. Layer model

always return to its activity [1]. Faults in the communication between MC and SC are considered transitory. The ”worker” components are not individually replicated-protected because their ”pool” structure allows implementing a task re-allocation in case of a worker fault. Failures of the networks are always considered as ”non-permanent” faults.

To protect the critical component (none-replicated) such as masters and cluster CM, the data and functions of these elements are replicated in certain workers. ”Replicas” of the critical components include the updated application, ”data structures” and their specific codes. These replicas are ”sleeping” and they are temporarily ”woken-up” for the updating of ”data” and ”control” values. Replicas are hosted in certain workers (background function) identified as protective workers (PWK).

The master information includes application data (problem and result data) and configuration data (a list of active workers and the list of assigned and remaining tasks).

The detection of an occurred failure is carried out through verifying the status of the cluster nodes. If any node of a cluster is inactive for a certain time (heartbeat), it is considered a failure node; the recuperation of this failure is then activated. In other words, ”inactivity” implies that there is a process monitoring (explicit or not) which can inform about the ”state” of the resource (machine or network).

The model is based on architecture having several workers with the same functionality and different data (Figure 3). The master has a copy of the data, and in the case of worker failure the running task will be reassigned to a different worker. The master has data from workers and monitors (watchdogs) the workers to detect their faults and to reallocate work. We consider the task of a worker in a data set as an atomic operation that works under a transactional scheme. If a task is allocated to

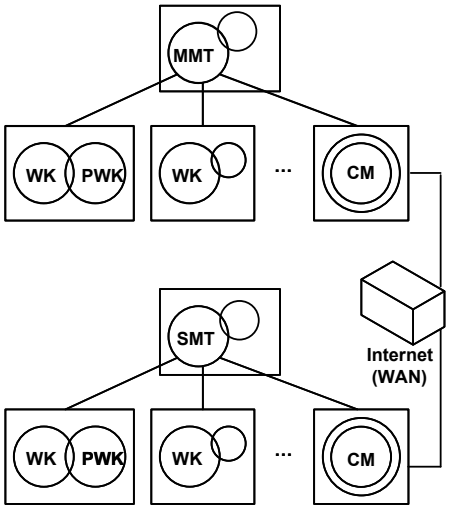

Figure 3. Logical functionalities

From	Action	To
MT	Replicate	PWKn/WKn and WKn
MT	Reconfigure	PWKn/WKn and WKn
MT	Protect	PWKn/WKn and WKn
MT	Protect	CM
MT	Reconfigure	CM
PWKn/WKn	Protect	MT
CM	Protect	SC

Figure 4. Protection scheme

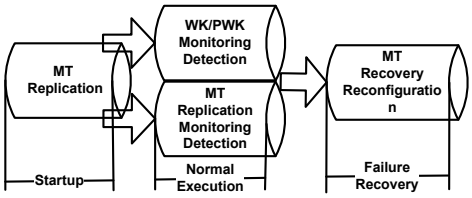

Figure 5. Fault tolerance activities

a worker, it runs and returns its results; another task can then be allocated to this worker. If it fails, the task will be allocated to other workers. MMT and SMT play the same role and use the same protection mechanism.

The MMT and SMT are responsible for dividing work into smaller tasks and for distributing tasks in an on-demand scheme to each worker. To solve the computational problem, we consider that the master communicates with the worker only to send a task and receive the results and that there is no communication between workers (transactional model). Our work considers a master-worker model where the master contains all initial application data and these data are divided into a certain amount of smaller data tasks, which in turn are also replicated.

The fault-tolerant scheme is summarized in Figure 4, where we show who is responsible (From) each one of the activities (Action), as well as the destination (To) of this activity.

There are three different phases for fault-tolerance activities in our model: startup, normal execution (replication, monitoring and detection (”protect”)), and failure recovery (reconfiguration) (Figure 5). Startup is the action of ”setup” for the cluster and in order to replicate the initial data. Normal execution represents prevention, including monitoring, fault detection and replication (protect). Fault recovery represents the action of recovering (i.e., so as to reconfigure) the cluster in the event of some of the foreseen faults in our model.

3.1. System Operation

After users define the level of protection, the three phases continue with their functions (Figure 5):

a) At startup time, the master information is replicated to certain workers. These special workers, called ”protective” workers (PWK), are those that can assume the master role in the event of master failure.

b) At normal execution without failures, before sending a task to a worker, the master multicasts this task index to all protective workers and to the target worker so that the PWK can maintain the consistency of master information. After completion of a task, each worker multicasts the result to the master and to each PWK. The PWK temporarily activates the ”sleeping master” to receive the ”worker-task results” and update the control information about the workers’ tasks. Therefore, the

master information is always updated in all ”C” PWK.

For detecting failures, certain messages are added to the original master-worker scheme: whenever a worker multicasts a result, the master and PWK send back a result-receipt acknowledgement to the worker; from time to time, each worker sends a heartbeat message to the master indicating that he is alive; from time to time the master also multicasts a heartbeat message to the PWK.

With these actions, it is possible to detect system failures. A master can detect any worker failure by counting the missing consecutive heartbeats for this worker. Whenever this number reaches a given limit, the worker is assumed to have failed. Workers can detect a master or PWK failure by not receiving their result acknowledgements, and PWK can also detect a master failure by not receiving a certain number of consecutive heartbeats.

c) In the event of any failure being detected, the system enters into the recovery phase (failure recovery). If failure in a worker is detected by the master, the master ascertains whether this worker is a basic WK, a CM or a PWK. A basic worker is simply dismissed from the system and all PWK are advised to modify their master information copy. The faulty worker's lost task will be reassigned to another worker (MT defines which worker, based on a priorities table). If the failed worker is a communication manager, the master not only acts by isolating the worker and updating the PWK state but also selects another worker to assume the CM function.

To solve a PWK failure, the master selects a basic worker to assume PWK functionality and sends all its information to this worker. It is of importance to observe that, while PWK failure is not recovered, there will be no work distribution and the workers will naturally be stopped, waiting for the failed PWK acknowledgement. The new PWK sends a special acknowledgement to the workers, and the workers update their multicast table.

Whenever a master failure is detected by one of the PWK, the PWK start a protocol for choosing the new master. This new master acknowledges the workers with this new information.

Figure 6 is a flow diagram showing a simplified model of the protection and recovery procedure interactions.

4. Validation

To validate and evaluate our proposal, we implement FT-DR, using as the test/benchmark program an MW version of the matrix multiplication (MM) algorithm [2] [5]. To evaluate the efficiency of the solution, experiments using an algorithm without FT-DR-and the same algorithm when modified-were performed, including FT-DR routines.

4.1. Results

The results of the experiment were obtained by using two geographically distributed clusters: one located at the Universidade Catlica do Salvador (UCSAL), Salvador, Bahia, Brazil and another located in the Universitat Autnoma de Barcelona (UAB), Barcelona, Spain,a heterogeneous environment. As a library for message passing, we use MPICH 1.2.5; an implementation supported in the standard MPI 1.1. The algorithms used in these experiments were compiled (gcc 3.2.3) and run on Kernel 2.4.20, the GNU/Linux Operating system. The LAN used in each one of the clusters is an Ethernet standard of 10 Mbps. In addition to UCSAL (worse condition), the connection with the Internet (WAN) was made through a non-dedicated link of 512 Kbps.

The results were obtained through this experiment (Figure 7) by means of executing the MM algorithm without FT-DR and the same algorithm when modified, including FT-DR routines (WK protection), at different time and days.

As the obtained data may show slight variation (order matrix multiplication of 1000, 1500 and

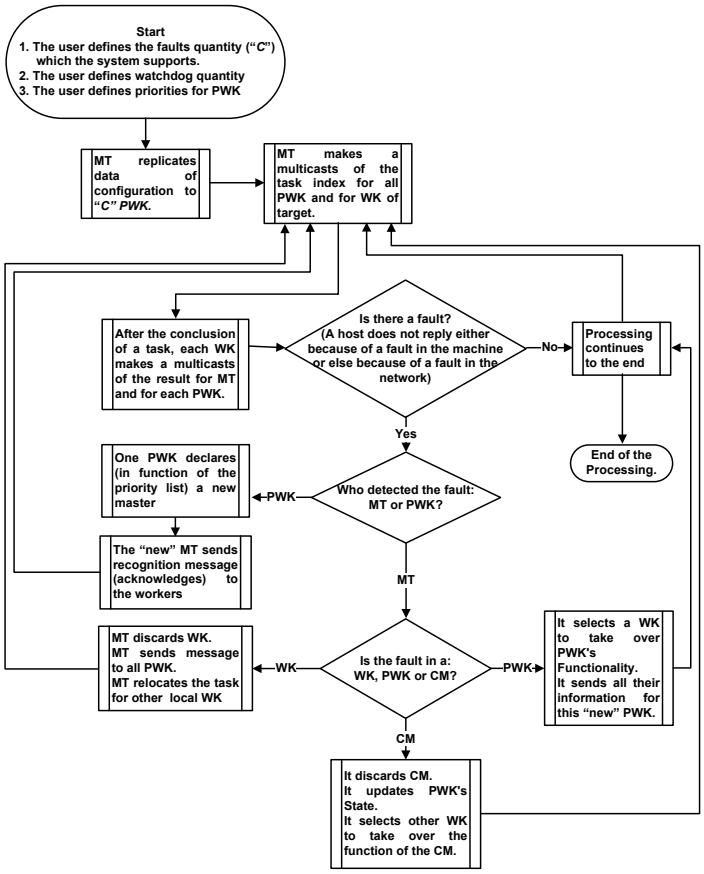

Figure 6. System operation

Matrix order	1000	1500	2000
Overhead	5,95%	2,02%	0,69%

Table 1
Overhead (%) after including the FTDR routines

2000 were used), we bear in mind that throughput and latency of connection is variable and influenced by the conditions of the network at the time of the experiments. As it is shown in Table 1, the insertion of FTDR routines (WK protection) in the MM algorithm, introduces an overhead (Execution Time with FTDR - Execution Time without FTDR) that decrease with high orders matrices.

5. Conclusions and further work

We have developed a dynamic replication scheme (FT-DR) for fault tolerance in wide-area clusters, for a common parallel programming paradigm, the Master/Worker.

Our FT-DR model in the preliminary experiments shows a rear-constant overhead and confirms the observations of Weissman [6] that replication-data model for a workstation environment is an excellent option for inserting Fault Tolerance into algorithms that solve great computational problems executed in CoHNOW environments.

Future work includes an analysis and evaluation of varying design options, in order to offer the

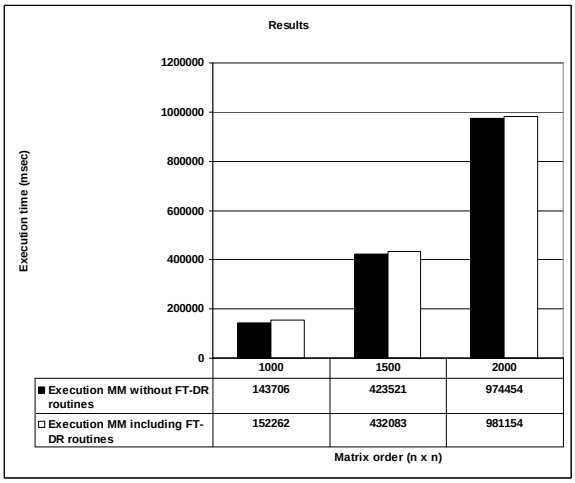

Figure 7. Execution Time and Overhead without and with FTDR routines

user different possibilities as regards both the application and user requirements.

Experimentation with additional MW applications is required. Future work will also include experimentation-critical elements, which our model (FT-DR) currently allows to fail.

References

[1] E. Argollo, J. R. Souza, D. Rexachs, and E. Luque. Efficient execution on long-distance geographically distributed dedicated clusters. *Lecture Notes in Computer Science*, v.3241:p.311–318, 2004.

[2] Olivier Beaumont, Vincent Boudet, Fabrice Rastello, and Yves Robert. Matrix multiplication on heterogeneous platforms. *IEEE Transactions on Parallel and Distributed Systems*, v12, n.10:p.1033–1051, 2001.

[3] George Bosilca, Aurelien Bouteiller, Franck Cappello, Samir Djilali, Gilles Fedak, Cecile Germain, Thomas Herault, Pierre Lemarinier, Oleg Lodygensky, Frederic Magniette, Vincent Neri, and Anton Selikhov. Mpich-v: Toward a scalable fault tolerant mpi for volatile nodes. *SC '02: Proceedings of the Proceedings of the IEEE/ACM SC2002 Conference*, page p.29, 2002.

[4] E. N. Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys, ACM Press*, v.34:p.375–408, 2002.

[5] A. Furtado, A. Rebouas, J. R. Souza, D. Rexachs, and E. Luque. Architectures for an efficient application execution in a collection of hnows. *Lecture Notes in Computer Science*, v.2474:p.450–460, 2002.

[6] Jon B. Weissman. Fault tolerant wide-area parallel computing. *Lecture Notes in Computer Science*, v.1800:p.1214–1225, 2000.